Snap!(BYOB) と Frama-C によるホーア論理入門 *

神戸大学人間発達環境学研究科
高橋　真

この資料はビジュアルプログラミング環境の Snap!(Build Your Own Blocks) 4.0[*1] と Frama-C[*2] を利用してホーア論理の初歩を分かりやすく解説することを目的としています.

* この講義資料は JSPS 科研費 23650507 及び 26560089 の助成を受けた研究の過程で得られたものです.

[*1] http://snap.berkeley.edu/

[*2] https://frama-c.com

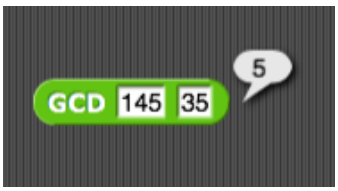
目次

1　Snap!プログラムの正当性

プログラムを作成するときに気をつけることというと，まず思いつくのは間違いのないプログラムを作成することであろう．間違いのないプログラムというときには，何をもって間違いがないと判断するのであろうか．通常は，何か目的をもってプログラムを作り，プログラムを動作させて必要な情報を手にいれるのである．従って，プログラムを動かしたときに，止まらなかったり[*3]或いは止まっても目的と違う結果が出てきたのでは困り，そのようなプログラムは“正しい”プログラムとは呼べない．プログラムが正しいとか間違っているとか言うときには，このように前もってプログラムに求めるものが決まっている．それをまとめたものを仕様 (specification) とよぶ．言い替えるとプログラムの正しさとは仕様に対して決まるのである[*4]．

『プログラムの正しさは仕様に対して決まる』とはどのようなことか次の Snap!のプログラムを例にとって考えよう．

このプログラムは，正整数 x と y の最大公約数をユークリッドの互除法により求めるモニターブロックである．変数 x と y の値を与えてブロックをクリックすると，変数 a の値が報告されそれが x と y の最大公約数になる．

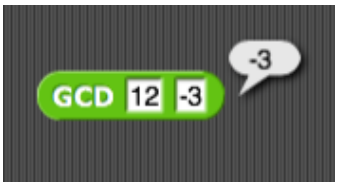


[*3] ここでは止まらないことが求められるプログラムは除外している．
[*4] この章の内容は文献 [1] を参考にしている．

問題 1.1　入力 $x = 145, y = 35$ に対し出力は 5 であることを変数表を書いて確かめよ．

ループ回数	0	1	2	3
a	145			
b	35			
c	0			

しかし，これで本当に最大公約数が求められているのであろうか．適当な値を x と y に与えると，確かに a の値は最大公約数になっているが“本当に”任意の値に対して最大公約数を求めているのか．変数 x と y に負の整数を与えたとき，上のプログラムはどういう結果を出力するだろうか．



上のプログラムは整数 x と y の最大公約数を求めるプログラムとすると間違いである．少なくとも一方が 0 ではない整数 x と y の最大公約数を求める正しいプログラムは

としなければならない．

最初のプログラムは，正整数に対して最大公約数を求めるという仕様に対しては正しいが，すべての整数に対して最大公約数を求めるという仕様に対しては正しくないのである．

Snap!プログラムで関数定義ブロックを入れて議論すると複雑になるので，この授業では次のような Snap!のプログラムに対しての“正しさ”について考察する．

1. 式とはモニター型のブロックで次のように再帰的に定義する．

2. ブール式は述語型のブロックで次のように再帰的に定義する．

3. プログラムはコマンド型のブロックで次のように再帰的に定義する．それぞれ代入文，複合文，条件文，until 文とよぶ．

定数や変数の定義がないが，任意の自然数を定数として扱え，変数は整数を値にもつもので Snap!で定義できるものとする．特に変数の値が整数であることに注意が必要な場合を除き，今後変数の値が整数であることは特に言及しない．

以上の Snap!プログラムの構成的な定義に従うと，Snap!プログラムが“正しい”ということを

証明方法として，構造に関する帰納法を適用することが考えられる．しかし，上で述べたようにプログラムが正しいとか正しくないというのは，仕様抜きでは考えられない．

そこでホーア論理とよばれるプログラムの正しさを検証するための論理学的方法が考えられた．ホーア論理では表明つきプログラムというものを使って，プログラムが仕様を満たすということを表現する．

A, B を何らかの条件を表わす（論理）式，P をプログラムとする．

$$\langle A\rangle\ P\ \langle B\rangle\ ,\quad \{A\}\ P\ \{B\}$$

のように表現されたものを表明つき Snap!プログラム，A を事前表明，B を事後表明とよぶ．事前表明事後表明をあわせて単に表明とよぶ．P の前後に表明をつけることで P に対する仕様を表現しているのである．仕様のわからないプログラムに対しては“正しい”とか“正しくない”ということは何も言えなかったが，表明つきプログラムに対しては定義可能である．$\langle A\rangle\ P\ \langle B\rangle$ ，$\{A\}\ P\ \{B\}$ が正しいということをそれぞれ次のように定義する．

定義 1.2

1. $\langle A\rangle\ P\ \langle B\rangle$ が正しいとは，A が成り立っているときにプログラム P を実行すると，その実行は停止し，さらに停止後 B が成り立つときをいう．
2. $\{A\}\ P\ \{B\}$ が正しいとは，A が成り立っているときにプログラム P を実行するとき，もし実行が停止するならば停止後 B が成り立つときをいう．（従って，P の実行が停止しない場合は $\{A\}\ P\ \{B\}$ は正しいと考える．）

前者にはプログラムの停止性が含まれているが，後者には含まれていない．$\langle A\rangle\ P\ \langle B\rangle$ を完全正当性，$\{A\}\ P\ \{B\}$ を部分正当性とよぶ．

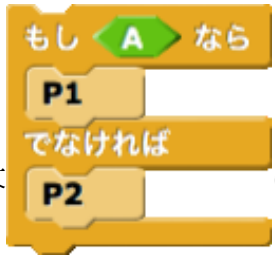
以上のように表明付き Snap!プログラム $\{A\}P\{B\}$ を定義することにより，それらが正しいことをプログラムの構成に関する帰納法で証明することが可能になる．具体的には，代入文の公理から複合文，条件文，until 文の推論規則及び帰結の推論規則を用いて証明をする．（Snap!プログラムのホーア論理の健全性：証明可能なものは正しい）

Snap!のプログラムをそのまま記載すると見にくいため，今後次のように Snap!のプログラムを略記することにする．

代入文 

は $x := t$， 複合文 

は $\mathrm{P}_1;\mathrm{P}_2$，

条件文 

は **if** A **then** P_1 **else** P_2 **fi**，until 文 

は **until** A **do** P **od** で表すことにし，必要に応じて Snap!プログラムを記載することにする．

また， 

は $A \wedge B$, 

は $A \vee B$, 

は $\neg A$ で表し，表明の表現にも同じ論理記号 $\wedge, \vee, \neg$ を用いることにする．

問題 1.3　次の Snap!プログラムを上で述べた略記法により記述せよ．

2　代入文の公理と帰結の規則

例 2.1　$\{y > 2\}\ x := y + 1\ \{x > 3\}$ は正しい.

代入文 $x := y + 1$ の実行前に変数 y の値が 2 より大きいとすると，代入文 $x := y + 1$ の実行により変数 x の値は $y + 1$ の値になるから，実行後の x の値は $2 + 1 = 3$ より大きくなる．従って，この表明付きプログラム $\{y > 2\}\ x := y + 1\ \{x > 3\}$ は正しい.

例 2.2　$\{x > 5\}\ \ x := x + 1\ \ \{x > 6\}$ は正しい.

代入文 $x := x + 1$ の実行前に変数 x の値が 5 より大きいとすると，代入文 $x := x + 1$ の実行により変数 x の値は 1 増加するから，実行後の x の値は 6 より大きくなる．従って，この表明付きプログラム $\{x > 5\}\ \ x := x + 1\ \ \{x > 6\}$ は正しい.

$R[x \to t]$ を表明 $R(x)$ の中にある変数 x を式 t ですべて置き換えて得られる表明とする．例えば，$R(x) \equiv [x > x^2 + 6]$ で $t \equiv x + 1$ のとき，$R[x \to t] \equiv [x + 1 > (x + 1)^2 + 6]$ である．このような表記法の下で，$\{R[x \to t]\}\ \ x := t\ \ \{R(x)\}$ は常に正しい表明つきプログラムである．この理由を考えてみる．代入文 $x := t$ の実行前に $R[x \to t]$ が成り立つとする．すなわち，変数 x の代わりに t を考えているので，表明 $R(x)$ 中の x の値を t の値にすると成り立つことになる．$x := t$ を実行すると，変数 x の値は t の値になるので，実行後に $R(x)$ が成り立つことになる．従って，$\{R[x \to t]\}\ \ x := t\ \ \{R(x)\}$ は常に正しい表明付きプログラムとなる．$\{R[x \to t]\}\ \ x := t\ \ \{R(x)\}$ の形の表明付きプログラムを代入文の公理とよぶことにする..

注意 $\{R(x)\}\ \ x := t\ \ \{R[x \to t]\}$ は正しい表明とはいえない.

例 2.3　$\{x < 5\}\ \ x := x + 1\ \ \{x + 1 < 5\}$ は正しくない.（x の値が 4 のときを考える.）

代入文の公理は，代入文 $x := t$ に関し，任意の事後表明 R に対しそれに適切な事前表明として $R[x \to t]$ がとれることを示している．もちろん，部分正当性の定義から $A \to R[x \to t]$ が正しい表明ならば，$\{A\}\ \ x := t\ \ \{R\}$ も正しい表明つきプログラムである.

例 2.4
- $\{x + 1 > 1\}\ \ x := x + 1\ \ \{x > 1\}$
- $\{x^2 > x^2 y\}\ \ x := x^2\ \ \{x > xy\}$

問題 2.5　次の代入文と事後表明に対し，代入文の公理となるように事前表明を定めよ.（式を計算せずに形式的に求める.）

	事前表明	代入文	事後表明
(a)	?	$x := x + 7$	$x + y > 20$
(b)	?	$x := x - 1$	$x^2 + 2x = 3$
(c)	?	$x := x - 1$	$(x + 1)(x - 1) = 0$
(d)	?	$y := x + y$	$y = x$
(e)	?	$y := x + y$	$y = x + y$

一般に $\{A\}\ P\ \{B\}$ が正しい表明つきプログラムで $C \to A, B \to D$ がともに正しい表明ならば，$\{C\}\ P\ \{D\}$ は正しい表明つきプログラムである．このことを次のような図式で表し，帰結の推論規則とよぶ.

帰結の推論規則

$$\frac{\{A\}\ P\ \{B\}}{\{C\}\ P\ \{D\}} \qquad \begin{array}{l}(\text{ただし } C \to A, B \to D \text{ が}\\ \text{ともに真のとき})\end{array}$$

この図式は，$C \to A$,$B \to D$ がともに真のときに，$\{A\}\ P\ \{B\}$ が正しければ，$\{C\}\ P\ \{D\}$ も正しいことを表す．今後，このようにいくつかの表明付きプログラムから別の表明付きプログラムを導く推論規則を定義するが，そこに現れる図式中の横棒は $\therefore$ と考えれば良い．また帰結の推論規則を用いるとき，$C \to A$,$B \to D$ が自明ではないときは，それが成り立つ理由を推論規則の横に記載することにする.

例 2.6 $\{x^2 > 2 \wedge x \in \mathbb{N}\}\ x := x - 1\ \{x > 0\}$ は正しい表明付きプログラムであることを，代入文の公理と帰結の推論規則を用いて示せ.

$$\frac{\{x - 1 > 0\}\ x := x - 1\ \{x > 0\}}{\{x^2 > 2 \wedge x \in \mathbb{N}\}\ x := x - 1\ \{x > 0\}} \qquad \begin{array}{l}(\because x^2 > 2 \text{ より，} x \neq 0, 1. \text{ よって，} x \in \mathbb{N}\\ \text{より } x > 1 \text{ であるから，} x - 1 > 0 \text{ である.}\\ )\end{array}$$

上式の $\{x - 1 > 0\}\ x := x - 1\ \{x > 0\}$ は代入文の公理であるから正しい表明つきプログラムで，下式の $\{x^2 > 2 \wedge x \in \mathbb{N}\}\ x := x - 1\ \{x > 0\}$ は帰結の規則により導かれている．よって，$\{x^2 > 2 \wedge x \in \mathbb{N}\}\ x := x - 1\ \{x > 0\}$ も正しい表明付きプログラムである.

問題 2.7 問題 2.5 で求めた事前表明を簡単な事前表明に書き換え，それを事前表明にもつ表明付きプログラムの正しさを代入文の公理と帰結の推論規則を用いて示せ.

3　複合文，条件文，until 文の部分正当性

3.1　複合文の部分正当性

次のような表明付きの複合文を考える．

$$\{a = 2 \wedge b = 3\}\ \ c := a;\ a := b;\ b := c\ \ \{a = 3 \wedge b = 2\}$$

複合文の実行は前から逐次的に行われるからこれは正しい表明付きプログラムである．我々は表明つきプログラムの正しさをその部分プログラムの正しさから導き出そうと考えている．したがって複合文においても，その部分プログラムの正しさから全体の正しさを導けるように考えたい．上記の例でそのことをみていこう．上の表明付きプログラムでそれぞれの代入文の実行前後で正しくなる表明を考えてみよう．まず，最初の代入文 $c := a$ では変数 c に変数 a の値が代入されるから，

$$\{a = 2 \wedge b = 3\}\ c := a\ \{c = 2 \wedge b = 3\}$$

は正しい表明つき Snap!プログラムである．さらに，これは代入文の公理の形をしていることに注意をしよう．同じように，

$$\{c = 2 \wedge b = 3\}\ a := b\ \{c = 2 \wedge a = 3\},\ \{c = 2 \wedge a = 3\}\ b := c\ \{b = 2 \wedge a = 3\}$$

も正しい表明つき Snap!プログラムである．最後の表明つき Snap!プログラムにおいて $b = 2 \wedge a = 3 \rightarrow$
$a = 3 \wedge b = 2$ が常に成り立つから，帰結の推論規則より事後表明を $a = 3 \wedge b = 2$ に変更してできる表明つき Snap!プログラム

$$\{c = 2 \wedge a = 3\}\ b := c\ \{a = 3 \wedge b = 2\}$$

も正しい表明つき Snap!プログラムである．これを逆に考えるとまず代入文の公理と帰結の推論規則から，

$$\{a = 2 \wedge b = 3\}\ c := a\ \{c = 2 \wedge b = 3\}$$
$$\{c = 2 \wedge b = 3\}\ a := b\ \{c = 2 \wedge a = 3\}$$
$$\{c = 2 \wedge a = 3\}\ b := c\ \{a = 3 \wedge b = 2\}$$

が正しいことがわかり，このことから最初の表明つき Snap!プログラム

$$\{a = 2 \wedge b = 3\}\ \ c := a;\ a := b;\ b := c\ \ \{a = 3 \wedge b = 2\}$$

の正しさが判断できるようになる．このように表明付きの複合文

$$\{A\}\ \ P_1; P_2; \cdots; P_n\ \ \{B\}$$

が与えられたときにその正しさを証明するには，新たに $n-1$ 個の表明 $S_1, S_2, \cdots S_{n-1}$ を適切に選んで

$$\{A\}\ P_1\ \{S_1\},\{S_1\}\ P_2\ \{S_2\},\cdots,\{S_{n-1}\}\ P_n\ \{B\}$$

という n 個の表明つき Snap!プログラムの正しさを示すことができれば

$\{A\}\ \ P_1; P_2; \cdots; P_n\ \ \{B\}$ の正しさを証明できることになる．このことをつぎのような推論規則で表すことにする．

$$\frac{\{A\}\ P_1\ \{S_1\}\ \{S_1\}\ P_2\ \{S_2\}\cdots\{S_{n-1}\}\ P_n\ \{B\}}{\{A\}\ \ P_1; P_2; \cdots; P_n\ \ \{B\}}$$

例題 3.1 $\{A\}\ \ x := x+1; y := x-y\ \ \{x < y\}$ が表明付きプログラムとして正しくなるように事前表明 A を定めよ．

証明： 代入文の公理より，$\{x < x-y\}\ y := x-y\ \{x < y\}$ は正しい表明付きプログラムで，同様に，$\{x+1 < x+1-y\}\ x := x+1\ \{x < x-y\}$ も正しい表明付きプログラムである．従って，A として $x+1 < x+1-y$ をとると，$\{A\}\ \ x := x+1; y := x-y\ \ \{x < y\}$ は正しい表明付きプログラムとなる．なお帰結の推論規則を用いて，A として，$y < 0$ をとってもよい．（こちらの方が適切ではある．） □

問題 3.2 次の複合文と事後表明に対し，表明付きプログラムとして正しくなるように事前表明を定めよ．

	事前表明	複合文	事後表明
(a)	?	$x := x-1; x := x^2$	$x > 0$
(b)	?	$x := x-y; y := x+y$	$x = 1 \wedge y = 2$
(c)	?	$x := x^2; y := x+y$	$x = 1 \wedge y = 2$

例題 3.3 $\{(x+y)^2 > x\}\ \ x := x+y; y := x-y; x := x^2\ \ \{x > y\}$ は正しい表明つき Snap!プログラムであることを代入文の公理，帰結の推論規則及び複合文の推論規則を用いて示せ．

証明： $A \equiv (x+y)^2 > x, S_1 \equiv x^2 > x-y, S_2 \equiv x^2 > y, B \equiv x > y, C \equiv (x+y)^2 > (x+y)-y$ とおく．$C \equiv S_1[x \to x+y], S_1 \equiv S_2[y \to x-y], S_2 \equiv B[x \to x^2]$ である．よって，

$$\frac{\dfrac{\{C\}\ x := x+y\ \{S_1\}}{\{A\}\ x := x+y\ \{S_1\}}\ \{S_1\}\ y := x-y\ \{S_2\}\ \{S_2\}\ x := x^2\ \{B\}}{\{A\}\ \ x := x+y; y := x-y; x := x^2\ \ \{B\}}$$

であるから与えられた表明つき Snap!プログラムは正しい[*5]． □

上の証明では代入文の公理に形をあわせるために帰結の推論規則が用いられたことに注意しよう．

[*5] このような図式を証明図 (proof figure) とよぶ．

問題 3.4 $\{xy+1>0\}\ x := xy; y := x+1; x := x+1\ \{x+y>0\}$ は正しい表明つき Snap! プログラムであることを代入文の公理，帰結の推論規則及び複合文の推論規則を用いて示せ．

3.2 条件文の部分正当性

次の表明付きの条件文を考える．

$$\{A\}\ \textbf{if}\ \alpha\ \textbf{then}\ P_1\ \textbf{else}\ P_2\ \textbf{fi}\ \{B\}$$

この条件文の実行はブール式 α が真であるか偽であるかにより決まる．α が真のときは P_1 が実行され α が偽のときは P_2 が実行される．従って，上の表明つき Snap!プログラムが正しいものであるためには，A が成り立っているとき，P_1, P_2 のいずれの実行においても B が成り立つことが必要である．すなわち，$\{A\wedge\alpha\}\ P_1\ \{B\}$,$\{A\wedge\neg\alpha\}\ P_2\ \{B\}$ が正しい表明つき Snap!プログラムのとき，$\{A\}$ **if** α **then** P_1 **else** P_2 **fi** $\{B\}$ が正しいと判断できる．このことを次のような推論規則を用いて表す．

$$\frac{\{A\wedge\alpha\}\ P_1\ \{B\}\quad \{A\wedge\neg\alpha\}\ P_2\ \{B\}}{\{A\}\ \textbf{if}\ \alpha\ \textbf{then}\ P_1\ \textbf{else}\ P_2\ \textbf{fi}\ \{B\}}$$

例題 3.5 $\{x=a\wedge y=b\}$ **if** $x\geq y$ **then** $z := x$ **else** $z := y$ **fi** $\{z=\max\{a,b\}\}$ は正しい表明つき Snap!プログラムであることを代入文の公理，帰結の推論規則及び条件文の推論規則を用いて示せ．

証明： $z=\max\{a,b\}$ を A で表す．

$$\frac{\dfrac{\{A[z\to x]\}\ z := x\ \{A\}}{\{x=a\wedge y=b\wedge x\geq y\}\ z := x\ \{A\}}\quad \dfrac{\{A[z\to y]\}\ z := y\ \{A\}}{\{x=a\wedge y=b\wedge\neg(x\geq y)\}\ z := y\ \{A\}}}{\{x=a\wedge y=b\}\ \textbf{if}\ x\geq y\ \textbf{then}\ z := x\ \textbf{else}\ z := y\ \textbf{fi}\ \{A\}}$$

$x=a\wedge y=b\wedge x\geq y$ ならば $\max\{a,b\}=a=x$ であるから，$A[x/z]$ が成り立つ．また，$x=a\wedge y=b\wedge\neg(x\geq y)$ ならば $\max\{a,b\}=b=y$ であるから，$A[y/z]$ が成り立つ．よって，最初の二つの推論は帰結の推論規則の条件をみたす．従って，与えられた表明つき Snap!プログラムは公理と推論規則で導かれるので，正しい表明つき Snap!プログラムである． □

問題 3.6 次の表明つき Snap!プログラム正しい表明つき Snap!プログラムであることを推論規則を用いて示せ．

1. $\{x>5\}$ **if** $x\leq y$ **then** $x := x$ **else** $c := x; x := y; y := c$ **fi** $\{x\leq y\}$
2. $\{x=a\}$ **if** $x<0$ **then** $x := -x$ **else** $x := x$ **fi** $\{x=|a|\}$
3. $\{y$ は自然数 $\wedge\, y>0\wedge z\cdot x^y=a\}$
   if $odd(y)$ **then** $z := z\cdot x; y := y-1$ **else** $x := x^2; y := y/2$ **fi**
   $\{y$ は自然数 $\wedge\, y\geq 0\wedge z\cdot x^y=a\}$
   ただし，$odd(y)$ は y が奇数であるかどうか判定する述語である．

3.3 Snap!プログラムと帰納的関数

前節の最後の問題で条件文に $odd(y)$ という述語やべき x^y を用いたが，このような述語や関数を用いることはSnap!プログラムの範囲を逸脱しないのだろうか．条件文に用いられるブール式に用いることのできる論理記号は $\wedge, \vee, \rightarrow, \neg$ のみで，$odd(y) \equiv (\exists x)_{x<y}[y = 2x+1]$ であるから[*6]，このままでは我々の定義したSnap!プログラムには使用できないように思える．また，べき x^y も我々の定義には含まれていない．しかし，実際には任意の帰納的関数 $f(x_1, \cdots, x_n)$ を用いて代入 $y := f(x_1, \cdots, x_n)$ を行ったり，任意の帰納的述語 $R(x_1, \cdots, x_n)$ を条件文のブール式として使用しても我々のSnap!プログラムの範疇からは逸脱しないことが示される．それは以下の定理によって保証される．

定理 3.7　任意の帰納的関数はSnap!プログラムにより計算できる．

この定理より $f(x_1, \cdots, x_n)$ が帰納的関数のとき $\{true\}\ P\ \{y = f(x_1, \cdots, x_n)\}$ を正しくするSnap!プログラム P が存在するので，プログラム中で $z := f(x_1, \cdots, x_n)$ となっているところを $P; z := y$ でおきかえると，もともとのSnap!プログラムの文法で書かれたプログラムになる．また，条件文で帰納的述語 $R(x_1, \cdots, x_n)$ を用いて，**if** R **then** P_1 **else** P_2 **fi** と使われている場合は，$R(x_1, \cdots, x_n) \Leftrightarrow f(x_1, \cdots, x_n) = 1$ となる帰納的関数 f が存在するので，この f を計算するSnap!プログラム P を用いて，（f の値が変数 y に入っているとすると），P; **if** $y = 1$ **then** P_1 **else** P_2 **fi** とするとやはりこれも，もともとのSnap!プログラムの文法で書かれたプログラムになる．以上のことから，代入文や条件文，until 文に帰納的な関数や述語を用いてもそれに対応するSnap!プログラムが存在するので，自由に使用してよいということになる．

3.4 until 文の部分正当性

次の表明つきSnap!プログラムを考える．

$$\{a = 0 \wedge b = 3\}\ \textbf{until}\ b = 0\ \textbf{do}\ a := a + 1; b := b - 1\ \textbf{od}\ \{a = 3 \wedge b = 0\}$$

$a = 0, b = 3$ で上の **until** 文を実行すると，until 文のループが回る間に変数 a, b は

ループ回数	0	1	2	3
a	0	1	2	3
b	3	2	1	0

となって確かに $a = 3, b = 0$ で止まるから，上の表明つきプログラムは（完全正当性の意味でも）正しい．

[*6] 従って，$odd(y)$ は原始帰納的述語である．

我々は表明つきプログラムの正しさをその部分プログラムの正しさから導き出そうと考えている．したがって until 文においても，until 文中の **do**$\cdots$**od** の間にあるプログラムの正しさから，全体の正しさを導くような規則が欲しい．つまり

$$\frac{\{A\}\ P\ \{B\}}{\{A'\}\ \textbf{until}\ \alpha\ \textbf{do}\ P\ \textbf{od}\ \{B'\}}$$

となるような推論規則がほしい．

従って，until 文のループの実行中にループの回る回数に無関係の条件があって，それが常に成り立っている状況が必要である．

このとき，A, B に求められるものは何であろうか．まず until 文で P が実行されるのは α が偽のときのみであるから，A が成り立つときには α も偽であるような条件でなければならない．従って A は“$\neg\alpha \wedge A_0$”のような表明でなければならない（あるいは意味がない）．

またループが 1 回実行されたとき，次にブール式 α の値を評価して偽になったときには，再び P を実行したいのであるから，B から A が導けなければならない．すなわち，少なくとも

$$\{\neg\alpha \wedge A_0\}\ P\ \{A_0\}$$

が正しいことが求められる．

このような A_0 はループの実行中に常に成立しているものなので，ループ不変表明と呼ばれる．

それでは上の例でこのような A_0 に対応するものは何であろうか．

ループ回数	0	1	2	3
a	0	1	2	3
b	3	2	1	0

ループの実行中に常に成立している関係を考えると $a+b=3 \wedge 0 \leq a \leq 3 \wedge 0 \leq b \leq 3$ である．a, b は整数であると仮定すると，$a+b=3 \wedge 0 \leq a \leq 3 \wedge 0 \leq b \leq 3 \Leftrightarrow a+b=3 \wedge a \geq 0 \wedge b \geq 0$ である．このとき，特に $a+b=3 \wedge\ \wedge b \geq 0$ について考えると

$$\{b \neq 0 \wedge (a+b=3 \wedge b \geq 0)\}\ a := a+1; b := b-1\{a+b=3 \wedge b \geq 0\}$$

は正しい．従って $a+b=3 \wedge b \geq 0$ はこの until 文のループ不変表明である[*7]．それでは，until 文の推論規則の上式を

$$\{\neg\alpha \wedge A_0\}\ P\ \{A_0\}$$

にしたとき，それからどのような表明つきの until 文が正しいといえるか．until 文を実行する前に A_0 が偽の場合は，部分正当性の意味から A_0 を事前条件に持つ表明つき Snap!プログラムはすべて正しい．A_0 が真のとき，もし α の値が真ならば until 文はなにもせずに終了するから，事後条件は事前条件から論理的に導かれるものならばよい．従って A_0 が真で，α が偽であるときが問

[*7] $a+b=3$ だけでもループ不変表明であるが，後で見るように全体での推論を考えると $a+b=3 \wedge b \geq 0$ にする必要がある．

題で，このときは上の式から α の値が偽である限りループが実行され，ループの停止後は α が真になる（このとき A_0 は真のままである）．そこで until 文の推論規則として，

$$\frac{\{\neg\alpha \wedge A_0\}\ P\ \{A_0\}}{\{A_0\}\ \textbf{until}\ \alpha\ \textbf{do}\ P\ \textbf{od}\ \{\alpha \wedge A_0\}}$$

を考えると確かに上式が正しければ，下式も正しくなる．$A \equiv a, b \in \mathbb{Z} \wedge a+b=3 \wedge b \geq 0$ として，

上の例に当てはめると

$$\frac{\{b \neq 0 \wedge A\}\ a := a+1; b := b-1\ \{A\}}{\{A\}\ \textbf{until}\ b=0\ \textbf{do}\ a := a+1; b := b-1\ \textbf{od}\ \{b=0 \wedge A\}}$$

となる．従ってこの推論規則を使って最初の表明つき Snap!プログラムの正しさを示すには次のような証明図を作成すればよい．ただし，$a, b \in \mathbb{Z}$ は省略している．

$$\dfrac{\dfrac{\left\{\begin{matrix}(a+1)+(b-1)=3\\ b-1\geq 0\end{matrix}\right\}\ a := a+1\ \left\{\begin{matrix}a+(b-1)=3\\ b-1\geq 0\end{matrix}\right\} \qquad \left\{\begin{matrix}a+(b-1)=3\\ b-1\geq 0\end{matrix}\right\}\ b := b-1\ \{A\}}{\left\{\begin{matrix}b\neq 0\\ A\end{matrix}\right\}\ a := a+1\ \left\{\begin{matrix}a+(b-1)=3\\ b-1\geq 0\end{matrix}\right\}}}{\{b \neq 0 \wedge A\}\ a := a+1; b := b-1\ \{A\}}$$

$$\frac{\{A\}\ \textbf{until}\ b=0\ \textbf{do}\ a := a+1; b := b-1\ \textbf{od}\ \{b=0 \wedge A\}}{\left\{\begin{matrix}a=0\\ b=3\end{matrix}\right\}\ \textbf{until}\ b=0\ \textbf{do}\ a := a+1; b := b-1\ \textbf{od}\ \left\{\begin{matrix}a=3\\ b=0\end{matrix}\right\}}$$

ここで，最上式の二つの表明つき Snap!プログラムはどちらも代入文の公理をみたし，帰結の規則を使用している 2 箇所は適用条件をみたしている．例えば，

$b \neq 0 \wedge A \rightarrow\ (a+1)+(b-1)=3 \wedge b-1 \geq 0$ は次のように示される．

$b \neq 0, a+b=3, b \geq 0$ とする．$a+b=(a+1)+(b-1)$ より，$(a+1)+(b-1)=3$ である．b は整数で，$b \neq 0, b \geq 0$ であるから，$b-1 \geq 0$ である．

問題 3.8

1. 次の表明つき Snap!プログラムは与えられた正整数 n と x の積を求めるプログラムである．

   $\{i=0 \wedge p=0 \wedge n>0\}$ **until** $i=n$ **do** $i := i+1; p := p+x$ **od** $\{p=n \cdot x\}$

   (a) $n=5$ の場合に変数 i と p がどのように変化していくか次の変数表に値を書いて正しい答えが得られることを確かめよ．また，$p=i \cdot x \wedge i \leqq n$ がループ不変表明になることを確かめよ．（$\top$ は真であることを表す．）

ループ回数	0	1	2	3	4	5
i	0					
p	0					
$p=i \cdot x$	$\top$					
$i \leqq n$	$\top$					

   (b) この表明付きプログラムの正しさを推論規則を用いて示せ．（変数が整数値をとることは仮定して良い．）

2. 次の表明つき Snap!プログラムは与えられた正の整数 a と b に対し，a を b で割ったときの商 $[a/b]$ と余り a **mod** b を求めるプログラムである．上の問題を参考にして適切なループ不変表明を定め，この表明付きプログラムの正しさを推論規則を用いて示せ．

   $\{a > 0 \wedge b > 0 \wedge q = 0 \wedge r = a\}$

   until $r < b$ **do** $q := q + 1; r := r - b$ **od**

   $\{q = [a/b] \wedge r = a \textbf{ mod } b\}$

3.5 帰結規則の使い方

ここまで複合文，条件文，until 文の推論規則を説明していくつか例をみてきた．その例をみるとわかるが表明つき Snap!プログラムが与えられたときそれを部分に分割して推論規則を適用していくステップは帰結の推論規則を除くと唯一に決まっている．すなわち，複合文には複合文の推論規則，条件文には条件文の推論規則そして until 文には until 文の推論規則が適用される．ただし，適用できるといっても until 文の推論規則を適用するためには，事前表明と事後表明の形に制限があるので，一般にはそのままでは適用できない．また，代入文の公理にも事前表明と事後表明の形に制限がある．それらを調整するために帰結の推論規則が使用される．

3.6 公理と推論規則のまとめ

ここまで考察してきた公理と推論規則をまとめると次のようになる．

代入文の公理

$$\{R[x \to t]\} \ \ x := t \ \ \{R\}$$

帰結の推論規則

$$\frac{\{A\} \ P \ \{B\}}{\{C\} \ P \ \{D\}} \qquad (\text{ただし } C \to A, B \to D \text{ がともに真のとき})$$

複合文の推論規則

$$\frac{\{A\} \ P_1 \ \{S_1\} \ \{S_1\} \ P_2 \ \{S_2\} \cdots \{S_{n-1}\} \ P_n \ \{B\}}{\{A\} \ \ P_1; P_2; \cdots; P_n \ \ \{B\}}$$

条件文の推論規則

$$\frac{\{A \wedge \alpha\} \ P_1 \ \{B\} \quad \{A \wedge \neg\alpha\} \ P_2 \ \{B\}}{\{A\} \textbf{ if } \alpha \textbf{ then } P_1 \textbf{ else } P_2 \textbf{ fi } \{B\}}$$

until 文の推論規則

$$\frac{\{\neg\alpha \wedge A\} \ P \ \{A\}}{\{A\} \textbf{ until } \alpha \textbf{ do } P \textbf{ od } \{\alpha \wedge A\}}$$

4　完全正当性

前章において部分正当性の意味で正しい表明つき Snap!プログラム導くための公理と推論規則について考察した．ここでは完全正当性について考えてみよう．部分正当性と完全正当性に違いはプログラムの停止性の保証があるかどうかである．部分正当性では停止性を保証する必要がなかったが，完全正当性ではプログラムの正しさばかりでなく停止性についても考慮しないといけない．代入文は必ず停止する．また前章で考察した部分正当性に関する推論規則のなかで until 文の推論規則を除くと，推論規則の上式が停止するプログラムならば下式は必ず停止することがわかる．一方 until 文の推論規則では，上式が停止するプログラムであっても下式が停止するとは限らない．例えば部分正当性の until 文に関する推論規則をそのまま適用して次のような完全正当性に関する推論を考えてみよう．

$$\frac{\langle 1 = 1 \wedge x > a \rangle\ x := x + 1\ \langle x > a \rangle}{\langle x > a \rangle\ \textbf{until}\ 1 \neq 1\ \textbf{do}\ x := x + 1\ \textbf{od}\ \langle 1 = 1 \wedge x > a \rangle}$$

明らかに上式は停止するが，下式は停止しない．

従って，部分正当性に関する until 文の推論規則を完全正当性に移し替えた

$$\frac{\langle \neg\alpha \wedge A \rangle\ P\ \langle A \rangle}{\langle A \rangle\ \textbf{until}\ \alpha\ \textbf{do}\ P\ \textbf{od}\ \langle \alpha \wedge A \rangle}$$

は上式の条件だけでは下式の停止性を保証できないため誤った推論であることになる．それではこの下式が（完全正当性の意味で）正しい表明つき Snap!プログラムであるためには上式の表明がどのような条件をみたしていればよいであろうか．ここで以前にとりあげた例をもう一度みてみよう．

$$\frac{\{b \neq 0 \wedge A\}\ \textbf{begin}\ a := a + 1; b := b - 1\ \textbf{end}\ \ \{A\}}{\{A\}\ \textbf{until}\ b = 0\ \textbf{do}\ a := a + 1; b := b - 1\ \textbf{od}\ \{b = 0 \wedge A\}}$$

この例では下式は b の値がどんな値であっても停止することがわかる．それは until 文の **do** 以下を実行するごとに b の値が 1 減るためにループを b 回くり返すと b の値が 0 になることがわかるからである．ここでは b がカウンターの役目をしていてループが回るごとに b の値がへり，やがて b の値が 0 になった時点で実行が終了するのである．

従って，一般の場合にもこの b に相当するカウンターがあってループが回るたびに減少してやがて 0 になるそのようなものがとれれば，停止性を示すことができる．そこで整数を値にもつ式 T で次の性質をみたすものを考える．

1. T はループの繰り返しで減少する．すなわち，$\langle \neg\alpha \wedge A \wedge v = T \rangle\ P\ \langle A \wedge\ T < v \rangle$ が正しい．
2. ループの繰り返しの間は T の値は正である．すなわち，$\neg\alpha \wedge A \rightarrow T > 0$ である．

もし，与えられた表明付きの until 文に対してこのような T を見つけることができれば，その

until 文の停止性を示すことができる．このような式 T を上界関数[*8]とよぶ．そこで次のような推論規則を考える．

$$\frac{\langle \neg\alpha \wedge A \wedge v = T\rangle\ P\ \langle A \wedge\ T < v\rangle}{\langle A\rangle\ \textbf{until}\ \alpha\ \textbf{do}\ P\ \textbf{od}\ \langle \alpha \wedge A\rangle}$$

（ただし，T は整数を値にもつ式で $\neg\alpha \wedge A \to T > 0$ をみたす）

この推論規則では，確かに上式が正しい表明つき Snap!プログラムならば下式も正しい表明つき Snap!プログラムである．これを用いることで多くの表明つき Snap!プログラムの正当性を示すことができる．例えば問題 3.8 において次の表明つき Snap!プログラムの正しさを証明したが，これの完全正当性版を考えてみよう．

$$\{i = 0 \wedge p = 0 \wedge n \geq 0\}\ \textbf{until}\ i = n\ \textbf{do}\ i := i + 1; p := p + x\ \textbf{od}\ \ \{p = n \cdot x\}$$

$T := n - i$ とおくと，この T が上界関数であることをわかるので，上記の推論規則より完全正当性の意味でも正しいことがわかる．

問題 4.1

$$\langle i \neq n \wedge p = i \cdot x \wedge i \leq n \wedge v = n - i\rangle\ i := i + 1; p := p + x\ \langle i \leqq n \wedge p = i \cdot x \wedge n - i < v\rangle$$

が正しいことを示せ．

問題 4.2

$$\langle i = 0 \wedge p = 1 \wedge n \geq 0\rangle\ \textbf{until}\ i = n\ \textbf{do}\ i := i + 1; p := p \cdot x\ \textbf{od}\ \ \langle p = x^n\rangle$$

が正しいことを示せ．（ヒント：上界関数としては，$T := n - i$ をとる．）

[*8] ここでは上界関数として整数を値にもつ式を用いたが，一般の場合には整礎集合上に値をもつ関数を用いる必要がある．

完全正当性に関する公理と推論規則をまとめると次のようになる.

完全正当性に関する公理と推論規則

代入文の公理

$$\langle R[t/x] \rangle \;\; x := t \;\; \langle R \rangle$$

帰結の推論規則

$$\frac{\langle A \rangle \; P \; \langle B \rangle}{\langle C \rangle \; P \; \langle D \rangle} \qquad (\text{ただし } C \to A, B \to D \text{ が ともに真のとき})$$

複合文の推論規則

$$\frac{\langle A \rangle \; P_1 \; \langle S_1 \rangle \; \langle S_1 \rangle \; P_2 \; \langle S_2 \rangle \cdots \langle S_{n-1} \rangle \; P_n \; \langle B \rangle}{\langle A \rangle \;\; P_1; P_2; \cdots; P_n \;\; \langle B \rangle}$$

条件文の推論規則

$$\frac{\langle A \wedge \alpha \rangle \; P_1 \; \langle B \rangle \quad \langle A \wedge \neg\alpha \rangle \; P_2 \; \langle B \rangle}{\langle A \rangle \; \textbf{if } \alpha \; \textbf{then } P_1 \; \textbf{else } P_2 \; \textbf{fi} \; \langle B \rangle}$$

until 文の推論規則

$$\frac{\langle \neg\alpha \wedge A \wedge v = T \rangle \; P \; \langle A \wedge \; T < v \rangle}{\langle A \rangle \; \textbf{until } \alpha \; \textbf{do } P \; \textbf{od} \; \langle \alpha \wedge A \rangle} \qquad (\text{ただし，} T \text{ は整数を値にもつ式で } \neg\alpha \wedge A \to T > 0 \text{ をみたす})$$

5 Frama-C による C プログラムの検証

5.1 Frama-C とは

Frama-C はフランスの 2 つの研究組織 CEA LIST (Software Security Laboratory) と INRIA Saclay - Ile-de-France (Toccata team, common with LRI-CNRS and Universitè Paris-Sud 11) により共同開発されている C プログラムの静的コード解析器である．様々な機能を持っているが，その中の一つに最弱事前表明を用いる WP プラグインがあり，ホーア論理的にプログラムの正当性を調べることができる．

5.2 仕様記述言語 ACSL

Frama-C では，プログラムの仕様を仕様記述言語である ACSL を用いて記述する．例えば，プログラムの事前表明は requires，事後表明は ensures，ループ不変表明は loop invariant などである．ホーア論理のところで演習を行った問題 3.8 を C 言語で記述して ACSL で記述すると次のように書くことができる．（C 言語には until はないので，#define で while を用いて定義している．）

```
#define until(exp) while(!(exp))
/*@      requires n > 0;
         ensures \result == n*x;
*/

int times(int n,int x){
         int i=0;
         int p=0;

/*@      loop invariant p==i*x;
         loop invariant i<=n;
         loop assigns p,i;
         loop variant n-i;
*/
         until (i == n){
                  p=p+x;
                  i=i+1;
         //@ assert p==i*x;
         }

return p;
}
```

表明は C のプログラムとしては，コメントとして扱われ，/*@と*/の間にあるもの，あるいは//@の後ろにある 1 行が Frama-C では表明として検証の対象になる．ensures にある \result は関数 times の返り値を表している．loop assigns は loop 中で値の変更される変数を指定するものである．また loop variant は停止性を証明するための上界関数を表す．assert はプログラム実行中に成立する性質を記述するものである．Frama-C ではこれらの条件がそれぞれ成り立つかを検証する．

5.3 Frama-C での検証例

次の表明付き代入文について Frama-C で検証する．

$$\{x > 5\} \;\; x := x + 1 \;\; \{x > 6\}$$

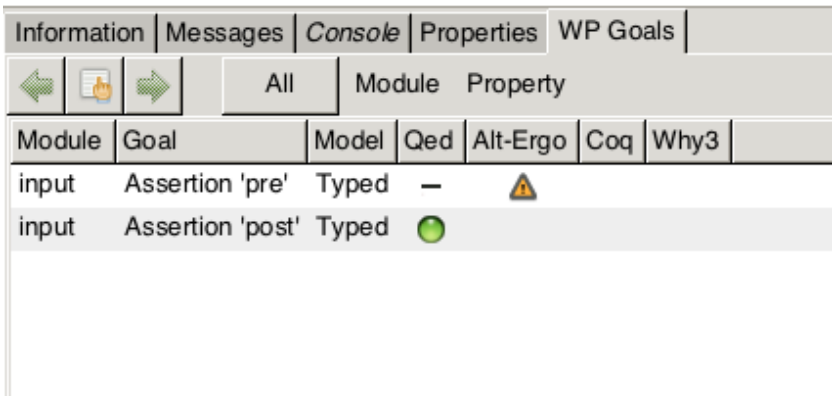
一つの代入文のみよりなる次の関数を考えて確かめる．

```
void input(void){
        int x;
        x++;
}
```

プログラムには次のように assert 文を入れる．

```
void input(void){
        int x=0;
        //@ assert pre: x > 5;
        x++;
        //@ assert post: x > 6;
}
```

asseret の後ろにある pre や post は表明に付けるラベルで，検証時にどの表明についての結果であるかわかりやくするために付ける．今の場合最初の assert 文は正しいとは限らないので，成り立つことを示すことはできないが，次の assert 文は最初の assert 文が正しいとしてチェックを行うので確かにホーア論理的な解釈で検証を行うことになる．Frama-C の GUI 版の frama-c-gui で検証すると次のように表示される．



条件文についての問題 3.6 の 1

$$\{x > 5\} \;\; \textbf{if } x \leq y \;\; \textbf{then } x := x \;\; \textbf{else } c := x; x := y; y := c \;\; \textbf{fi} \;\; \{x \leq y\}$$

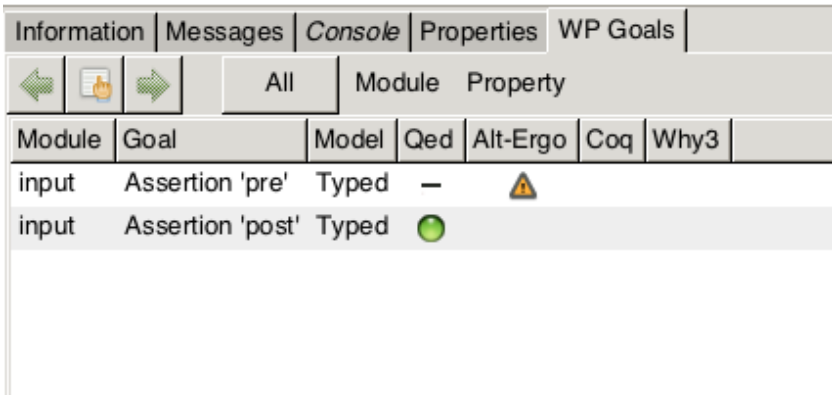
を同様に次のような関数で検証する．

```
void input(void){
        int x,y,c;
```

```
        //@ assert pre:x > 5;
        if (x<=y) x=x;
        else{
                c=x;
                x=y;
                y=c;
         }
        //@ assert post:x<=y;

}
```

このときも確かに



となる.

5.2 で最初に挙げた問題 3.8 を C 言語で記述したものも frama-c で検証するとすべて valid となる.

参考文献

[1] プログラム検証論　林晋著　共立出版

[2] コンピュータサイエンス入門　論理とプログラム意味論

田辺誠・中島玲二・長谷川真人著　岩波書店

[3] プログラム仕様記述論　荒木啓二郎・張漢明著　オーム社

[4] Frama-C　https://frama-c.com/